VERTEX COLORING ALGORITHMS

Pallavi Sharma
Department of EECS
Washington State University

ABSTRACT

The Vertex Coloring Problem (VCP) aims to find the k-coloring of Graph $G(V, E)$ such that the value of k is minimized. This paper focuses on the understanding and solving the Vertex Coloring Problem (VCP). Both exact and heuristic approaches have been discussed along with their analysis. The paper also describes the applications of the VCP along with the recent advances in this area.

1 INTRODUCTION

The vertex coloring problem (VCP) has been an intriguing problem since the year 1852. It started with the problem of coloring the countries in a map such the countries that share a common border are not colored with the same color. Mathematically, the same problem can be translated into a planar graph with the countries denoted as points or vertices and lines or edges being used to connect the points that share a common border. In 1879, Alfred Kempe proved that any planar graph can be colored with 4 colors, this is also called as the Four Color Problem. However, in 1890, Percy John Heawood proved that 5 colors are sufficient to color any map. Thus, this Four Color Problem became the starting point for research on the Coloring theory among the mathematicians. Mathematicians took different directions to solve this problem including algebraic and combinatorial methods. Finally, in 1976 through the extensive use of computers, Appel, Haken and Koch proved that four colors are enough to color any map. In 1912, George David Birkhoff proposed on using chromatic polynomials for studying graph coloring problems and this initiated research on hypergraphs and chromatic spectrums. Even today exact and heuristic approaches are being developed for numerous graph coloring problems. This paper aims to discuss this problem in detail, including the exact and heuristic algorithms that can be used to solve this problem and the real-world problems which can be solved through these approaches.

2 DEFINITIONS

Let us now formally define the vertex coloring problem and terminology related with it.
The vertex color problem for an undirected graph G with V vertices and E edges $(G = (V, E))$ is defined as the assignment of a color to each vertex of the graph G such that no two adjacent vertices receive the same color and the number of distinct colors used in this process is minimum. This classic problem can also be referred as the k-coloring of graph G where k denotes the number of colors used to assign the vertices.
An independent set is defined as any subset of vertices (V) that doesn't contain any adjacent vertices. This implies that the k-coloring of graph G is equivalent to the partition of V into k independent sets. Each independent set in this case would represent a distinct coloring and hence is called a color class.
The k-coloring of the graph G using the smallest possible value of k is termed as the optimal coloring of the graph G and the value of k used for this coloring is used to represent the chromatic number of graph G $(\chi(G))$
Let us apply the coloring theory to the graph G shown in figure 1. This graph is a cycle graph C_5. We can color this graph by assigning a distinct color to each vertex making it a 5-coloring, however this is not the optimal coloring of graph G. The graph can be colored using colors fewer than 5. Considering that each vertex has degree 2 i.e. it has two adjacent vertices, this can be achieved using three colors as shown in Figure 2. Hence, we can conclude that G is a 3-colorable graph. The color

classes or independent sets for G are: $\{A, D\}$, $\{B, E\}$ and $\{C\}$ and finally, the chromatic number $\chi(G)$ is 3.

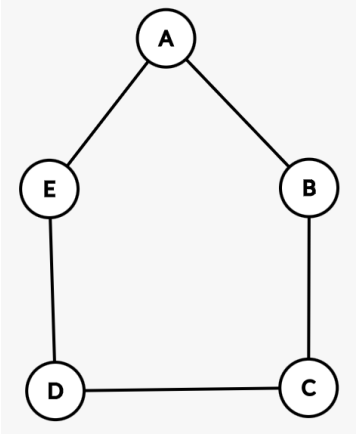

Figure 1: $G = C_5$

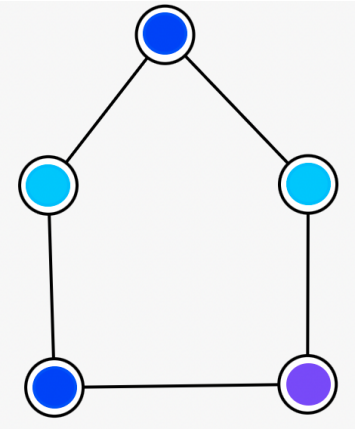

Figure 2: k-coloring of G

3 MODELS

This section covers the commonly used integer linear programming models (ILP) for the vertex-coloring problem. Most of the exact and heuristic algorithms are based on these models.

3.1 VCP-ASS

This model assumes that n colors would be sufficient for coloring graph G. The ILP equations written using this model are primarily based on two variables x_{ih} and y_h. The variable x_{ih} in general specifies the color assigned to each vertex, here $i \in V$ and $h \in \{1, 2, 3, 4..., n\}$, hence, $x_{ih} = 1$, if and only if vertex i is colored with color h and $y_h = 1$ if and only if color h is being used in the solving the problem. The set of equations that define this problem are as follows:

$$\min \sum_{h=1}^{n} y_h \tag{1}$$

$$\sum_{h=1}^{n} x_{ih} \; where \;\; i \in V \tag{2}$$

$$x_{ih} + x_{jh} \leq y_h \; where \; (i, j) \in E, h \in \{1, 2, 3, .., n\} \tag{3}$$

$$x_{ih} \in \{0, 1\} \; where \;\; i \in V, h \in \{1, 2, 3, .., n\} \tag{4}$$

$$y_h \in \{0, 1\} \; where \;\; h \in \{1, 2, 3, .., n\} \tag{5}$$

The objective function is represented in equation (1) and it tries to minimize the number of colors used in coloring the graph. The equation (2) makes sure that each vertex is colored whereas equation (3) ensures that at most one pair of adjacent vertices, represented by $(i, j) \in E$ should receive the same color. For all the other adjacent pair of vertices, for a given color h, the inequality $x_{ih} + x_{jh} < y_h$ must hold true. The constraints (4) and (5) have been imposed to make sure that the variables x_{ih} and y_h contain only binary values i.e. $\{0, 1\}$

Use of binary variables and presence of polynomial number of variables make this model easy to extend and use for many exact and heuristic approaches. However, a major drawback of the model is the absence of a unique solution. If the solution obtained from solving this model consists of k colors, there is no way to distinguish which k colors out of the total n colors should be used. There exists $\binom{n}{k}$ ways to select k colors from n and these k colors can be permuted in $k!$, leading to a total of $\binom{n}{k}$ $k!$ equivalent solutions.

3.2 VCP-SC

This model is based on the independent sets of the graph. Assume S to be the family of all the independent sets of graph G. Each independent set $s \in S$ has a variable x_s such that $x_s = 1$ if and only if all the vertices present in the set s are assigned the same color. The set of equations that define this problem are as follows:

$$\min \sum_{s \in S} x_s \tag{6}$$

$$\sum_{s \in S : i \in s} x_s \; where \;\; i \in V \tag{7}$$

$$x_s \in \{0, 1\} \; where \;\; s \in S \tag{8}$$

The objective function for this model is represented by equation (6) and it tries to minimize the count of the independent sets or in other words the color classes of the graph G and hence, is trying to reduce the number of colors used. The equation (7) imposes constraint on the number of sets a vertex can belong to. Since, each vertex should be a part of at-least one independent set, this value must be greater than or equal to one. In cases, when the vertex belongs to more than one set, a feasible solution can be computed by keeping the vertex in one set and removing it's occurence from all the others. The equation (8) makes sure that the variable x_s takes only binary values.
The biggest advantage of this model is that, unlike the VCP-ASS model it is invariant to the colors chosen and hence, would always lead to a unique solution.

4 DSATUR ALGORITHM

The DSATUR ALGORITHM is based on the ILP model VCP-ASS and is an exact algorithm for finding the optimal solution for k-coloring. The algorithm is one of the most popular sequential vertex coloring algorithm. This algorithm solves the drawback of $\binom{n}{k}\ k!$ equivalent solutions presented by the VCP-ASS model by determining a fixed sequence of the vertices, thereby resulting in a unique solution. In this section we, will be discussing the original algorithm along the variations.

4.1 General Algorithm

The algorithm consists of two parts- one relates to strict ordering of the vertices present within Graph G and other consists of choosing a vertex selection rule. The rules governing the next vertex which must be assigned a color is called a Vertex Selection rule. Since this algorithm is recursive, in order to reduce the complexity, only tight colorings are considered.
Let the graph G has n vertices and m edges. A coloring C of a the graph G is tight, given a strict ordering of vertices $v_1, v_2, ...v_n$ if

$$c(v_1) = 1 \tag{9}$$

$$colors(i) = max_s c v_s \; where \; 1 \leq s \leq i \; and \; i \in \{1, 2, .., n\} \tag{10}$$

$$c(v_{i+1}) \leq colors(i) + 1 \; \forall \; i \in \{1, 2, .., n\} \tag{11}$$

Since the cardinality of V is n. The available colors will be labelled as a set containing all natural numbers from 1 till n.

$$Colors = \{1, 2, 3, ..., n\} \tag{12}$$

The general algorithm is as follows:

1. The uncolored vertex v is selected using a vertex selection rule r (these are explained in the sections below)
2. A color i is assigned to the vertex v such that i has the smallest value in the set $Colors$ and this value has not been assigned to any of it's neighbors.
3. The above process is repeated until there is no uncolored vertex left.

The sections 4.2, 4.3 and 4.4 define the various vertex selection rules that can be used for this algorithm.

4.2 MAXIMUM SATURATION DEGREE

The saturation degree for a vertex is calculated by counting number of distinct colors assigned to the adjacent vertices. In this rule, the uncolored vertex with the maximum degree of saturation is selected. In case of ties, the vertex with maximum degree is selected. If the tie persists, the vertex ahead in the strict ordering is chosen. This heurtistic rule is used when solving a problem using the original DSATUR algorithm.

4.3 SEWELL RULE

The vertex with the maximum saturation degree is selected. In case of ties, the vertex with maximum number of common available colors in the neighborhood of uncolored vertices is selected.
The function $same$ is calculated for all the uncolored vertices present at a given time. The definition of the function is as follows:

$$same(v_1, v_2) = \begin{cases} 0 & \text{if } (v_1, v_2) \notin E \\ |F(v_1) \cap F(v_2)| & \text{if } (v_1, v_2) \in E \end{cases} \tag{13}$$

where $F(v_i)$ is the set of allowed color labels for a uncolored vertex v_i We also observe that the vertices don't contribute to the count if they are not joined by an edge. However, if they share an edge, the function returns the intersection of the sets containing allowed colors for both the vertices. Let U be the set of the uncolored vertices at the current step and T be the set of candidates that possess the same maximum saturation degree i.e. $T \subseteq U$. The number of allowed colors is calculated for all the candidates in T and the expression used for calculating the selected vertex (v_{sel}) is as follows:

$$v_{sel} = max_{v_s \in T} \, (\sum_{v \in U, v \neq v_s} same(v_s, v)) \tag{14}$$

We can observe that although the resulting vertex is selected from the candidate set T, the $same$ function for each of these candidates is calculated with respect to all the vertices present in the set U i.e. all uncolored vertices. Hence, although the rule tries to reduce the search space and motivates using the common colors, is expensive to compute.

4.4 PASS RULE

This rule also selects the vertex with the highest value of saturation degree. It's tie-breaking rule is similar to Sewell rule but is much less expensive in computation. The first difference is that the rule is applied only at the steps where the number of allowed colors for the vertices in the set T denoted by μ is below the parameter TH and this value is usually taken as 3. Secondly, when computing the v_{sel}, the $same$ function is computed only with respect to the vertices present in set T and not set U.

$$v_{sel} = max_{v_s \in T} \, (\sum_{v \in T, v \neq v_s} same(v_s, v)) \tag{15}$$

In a nutshell, the algorithm follows the below process:

$$rule\ applied\ at\ each\ step\ in\ case\ of\ ties = \begin{cases} PASS\ rule & \text{if } \mu \leq TH \\ DSATUR\ rule & \text{if } \mu > TH \end{cases} \tag{16}$$

4.5 ANALYSIS OF THE ALGORITHM

For any partial k-coloring at a given step, the vertex is chosen using the rules mentioned above, each vertex has the choice of either selcting one of the k colors that have already been used before or choose the k+1 color if these colors are not available. Hence, at any step of this algorithm, each vertex has (k+1) colors to select from. Thus, the search tree achieved while running this algorithm has a maximum branching factor of (k+1).
Another point worth noticing is that when we reach a leaf node while traversing the search tree, the size of the coloring helps in updating the upper bound (UB) for the chromatic number of G. If the size is smaller than the UB, the UB is updated with the size of current coloring. This process helps in pruning the search tree and reducing the time taken to run the algorithm.

5 HEURISTIC ALGORITHMS

The exponential running time of the exact algorithms prompted the scientists and researchers to find heuristic algorithms. Greedy coloring is one form of heuristics that is often used in this area. It involves selection of vertices one after the other in a sequence, based on a rule and unlike exact algorithms involving creation of search tree, once a vertex is colored, it cannot change it's color. These algorithms are very fast and are highly sensitive to the ordering of the vertices.

5.1 FIRST FIT (FF)

Assume the vertices in graph G be assigned labels $v_1, v_2, ...v_n$ and the colors palette be labelled as $1, 2, 3, .., n$. The process begins with labeling the vertex v_1 with the lowest value of the colors available ensuring that the adjacent vertices don't receive the same color. Since, v_1 is the first vertex, it is assigned the color 1 and this assignment stands fixed and cannot be updated in future. The same process is continued for all the vertices from v_2 till v_n. The generic process for a vertex v_i can be expressed as follows:

$$v_i = min(colors)\ s.t.\ color \notin N(v_i) \tag{17}$$

$$where\ 1 \leq i \leq n \tag{18}$$

$N(v_i)$ refers to the neighbors of the vertex v_i i.e. the vertices joined with vertex v_i by an edge.

5.2 LARGEST DEGREE ORDERING (LDO)

This algorithm selects and assigns a color to the vertex having the maximum neighbors or in other words, the one with the highest degree, making sure that none of the neighboring vertices receive the same color. The process to select the vertex with largest degree continues till all the vertices are colored. Since, the algorithm focuses on targeting the vertices that are connected to maximum number of vertices first, it should produce a better coloring than the First Fit method.

5.3 INCIDENCE DEGREE ORDERING (IDO)

The incidence degree of a vertex is defined as the number of it's colored neighbors. The algorithm selects the vertex with the largest incidence degree and assigns a color with the lowest value such that the neighboring colored vertices do not possess the same color. The same process can be depicted as follows:
Let U be the set of uncolored vertices. Initially, the $U = V$ and as the vertices get selected and colored, the set U reduces in size

$$v_{next} = max_{v_i \in V}(IncidenceDegree(v_i)) \tag{19}$$

$$color(v_{next}) = min(colors)\ s.t.\ color \notin N(v_{next}) \tag{20}$$

$$U = U - v_{next} \tag{21}$$

The above process is repeated till the set U becomes empty.

5.4 SATURATION DEGREE ORDERING (SDO)

The Saturation Degree, as defined in the section above is referred as the number of adjacent vertices that are distinctly colored.This ordering follows the same process as defined above for the Incidence degree and selects the vertices with maximum degree of saturation.

5.5 COMBINATION OF LDO AND IDO

This algorithm uses a combination of two orderings. While the algorithm remains the same as the above algorithms, the selection process has become more robust. Similar to LDO, the vertex with maximum degree (deg) is selected, in case, there is more than one vertex with the largest degree, the vertex with maximum value of incidence degree (ID) is selected and assigned a color.

$$v_{next} = \begin{cases} max_{v_i \in V}(deg(v_i)) & \text{if } single\ vertex\ exists\ with\ maximum\ value \\ max_{v_i \in K}(ID(v_i)) & \text{if } two\ or\ more\ vertices\ exist\ with\ same\ deg\ represented\ by\ set\ K \\ selected\ at\ random\ from\ K & \text{if } one\ or\ more\ vertices\ exist\ with\ same\ ID \end{cases} \tag{22}$$

5.6 COMBINATION OF SDO AND LDO

This algorithm uses a combination of the SDO and LDO to select next vertex for color assignment. The vertex is selected first on the basis of highest saturation degree (SD). The ties are broken by selecting the one with maximum degree(deg). The equation below explains the process of vertex selection:

$$v_{next} = \begin{cases} max_{v_i \in V}(SD(v_i)) & \text{if } single\ vertex\ exists\ with\ maximum\ value \\ max_{v_i \in K}(deg(v_i)) & \text{if } two\ or\ more\ vertices\ exist\ with\ same\ SD\ represented\ by\ set\ K \\ selected\ at\ random\ from\ K & \text{if } one\ or\ more\ vertices\ exist\ with\ same\ deg \end{cases} \tag{23}$$

6 OBSERVATIONS AND INFERENCES

After analyzing the ILP models, DSATUR algorithm and it's variants and heuristic algorithms for vertex coloring, there are some key observations worth noting:

1. Based on the exact algorithms and ILP models studied, the vertex coloring problem is computationally expensive to solve even for graphs with around 100 vertices. The recursive algorithms compute many partial k-colorings at each step i.e. for each vertex before reaching the optimal solution. This requires enormous time and memory, making it difficult to solve for large graphs. The same is true for ILP models, where the constraint equations grow larger in number and are difficult to compute using ILP solvers. Thereby, justifying that this problem belongs to the class of NP-Hard problems.
2. The hardness of the exact VCP computation has led to the development of heuristic algorithms. These algorithms are usually highly sensitive to certain parameters such as initial ordering of the vertices. The heuristic algorithms also don't calculate multiple colorings at each step, once the color is assigned to a vertex, it stays fixed. Unlike the exact approaches, which compare many feasible solutions to obtain the optimal solution, this approach results in a single solution which is not guaranteed to be the optimal coloring. This method has to be repeated multiple times using different parameters to in order to determine the optimal solution.
3. The choice of algorithm for a graph depends on the size (m) and density of the graph ($\frac{m*100}{m_{max}}$). The DSATUR algorithm is computationally more effective than the brute-force approach by using a heuristic rule for selection of the vertices and discarding the feasible solutions that use more colorings than the upper bound (UB). This helps in reducing the search space and hence makes the algorithm run faster. However, this algorithm would still be effective for small graphs with up to 70 vertices. Heuristic algorithms on the other hand, are both easier and faster to implement and hence work well for large graphs with more than 1000 vertices.
4. The exact algorithms actually use heuristics to reduce the complexity of the problem. The similarity between the methods lies in using vertex selection rules for choosing the next vertex to be colored and the difference lies in how the color is assigned. While the exact algorithms, considers solutions with the selected vertex being assigned all the different available colors, the heuristic algorithms usually assign the vertex with allowed color of minimum value.
5. From [4], when DSATUR, SEWELL and PASS algorithms were applied to random graphs with number of vertices ranging from 60 till 1000. The PASS algorithm outperformed both the algorithms in terms of time and accuracy. This is intuitive, since the PASS rule variation of DSATUR algorithm applies both the original heuristic as well as the PASS rule at different steps based on a threshold. Initially, the algorithm follows the DSATUR rule and once, the vertices begin receiving colors, the selection rule switches to PASS rule. This helps in reducing the time taken to run the algorithm and hence, makes the algorithm faster than the other two.
6. When analyzing the heuristic algorithms, the FF algorithm is the simplest and fastest to run. However, the algorithm lacks any robust strategy for selection and the resulting coloring is extremely dependent on the ordering of the vertices and may not be optimal. The rest of

degree selection algorithms are less sensitive to the ordering of the vertices as the vertex is selected based on the maximum value of the degree. The combination algorithms LDO + IDO and SDO + LDO define a more strict criteria by providing a tie-breaking strategy and hence, reduce the probability of selecting any vertex at random. The empirical evidence supports the above theory as the combination algorithms produced a more optimal coloring over the others when tested for random graphs with number of vertices ranging from 200 till 1000[5].

7 APPLICATIONS

The applications of VCP in the real-world are enormous. A few of these applications are discussed below:

7.1 EXAM SCHEDULING

Often times, coordinators struggle to come up with the exam schedule due to the multiple students enrolling for different courses, making it hard to run parallel exams. The idea is to schedule the exams such that none of the exams overlap for the students who are enrolled in these courses and the number of days required to conduct the exams is minimum. This is clearly a vertex coloring problem, where the courses act as vertices and they are joined by edges if there exists students that have enrolled for both the courses. The resulting graph can be colored and the courses with the same color can have exams scheduled on the same day. The same has been illustrated in figure 3 and 4. On the applying VCP algorithms to graph shown in figure 3, exams for the courses with the same color i.e. Chemistry, Physics and CS can be scheduled on one day and for courses English and Maths on the next day.

7.2 MAP COLORING AND MOBILE NETWORKS

The vertex coloring algorithm is used to color geographical maps such that the neighboring regions receive different colors. This helps in dividing a country or region into distinct geographical areas. Since, the maps are planar graphs, according to the four-color theorem, they can be colored using at most 4 colors. A dual graphs is created for the map by placing a vertex inside each region and joining vertices with edges if the regions share a common boundary. This application is useful for the Groups Special Mobile (GSM) networks. GSM Network is a mobile phone network that divides the geographical area of the network into hexagonal cells or regions and mobile phones get connected to the GSM network by searching for neighboring cells. Since GSM can only operate at 4 different frequencies, by applying vertex coloring on the regions of the map using 4 distinct colors can help in efficiently assigning the frequencies to the regions.

7.3 TIMETABLING

Similar to exam scheduling, when creating university or school timetables, classes for different subjects need to be allotted different times and days in order to avoid conflicts. This is a vertex coloring problem where the subjects act as vertices and they are joined with an edge if they share common students. The objective is to color the resulting graph such that the number of time slots required to schedule these classes is minimim. The courses assigned the same color are independent and can have overlapping time slots whereas distinct colors demand non-overlapping time slots. This problem can be further extended by considering the availability of instructors.

7.4 AIRCRAFT SCHEDULING

Aircraft scheduling is usually done by airlines based on the availability of aircrafts and number of flights scheduled. This can be translated into a vertex coloring problem where the aircrafts are considered as vertices and an edge exists between two vertices if their flight times coincide or overlap. Vertices need to be colored such that time interval within which all the flights are scheduled is minimized. The resulting graph can be used to assign the aircrafts with the flights they will be handling.

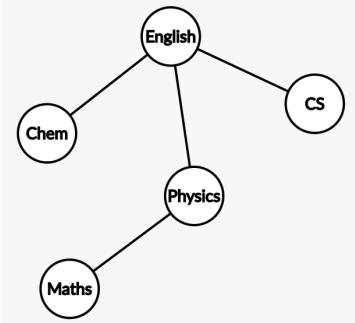

Figure 3: Graph created using courses

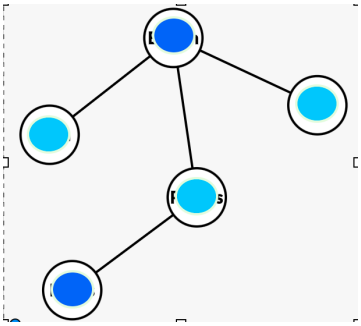

Figure 4: 2-coloring of graph

8 Future Scope

VCP is one of the most classical problems of graph theory. This problem and the devised approaches became the starting point of further research and development, some of which can be can be found below:

1. VCP has many variants. The most popular one being the Weighted Vertex Coloring Problem (WVCP) or max-coloring problem. Given a graph $G(V, E)$ with each vertex $v \in V$ having an associated weight given by $w(v)$, the task is to find a coloring $\{V_1, V_2, V_3, ..V_k\}$ of G such that $\sum_{i=1}^{k} max_{v \in V_i} w(v)$ is minimized. Taking $w(v) = 1 \ \forall \ v \in V$ translates WVCP into VCP. Both exact and heuristic algorithms exist for this problem.
2. Another variant that emerged was the minimum sum coloring problem (MSCP). This problem aims to find the k-coloring of the graph such that the sum of colors assigned to the vertices is minimum. The objective function can be written as follows:

$$\sum_{v \in V} c(v) \tag{24}$$

   where $c(v)$ represents the value of color assigned to vertex v.
3. Research today is focusing on using combination of metaheuristic or heuristic approaches to improve the computation time for VCP. For instance, recently in 2021, Bruno [10] established how the iterated local search with tabu search can be used for WVCP.
4. VCP has also been transformed into equivalent problems in other domain areas, for instance genetics and oscillatory networks [11]. In [11], the undirected graph (G) is converted into a circuit containing battery, resistances and capacitors.
5. The idea of VCP has also been extended to hypergraphs. A hypergraph is defined as a graph in which hyperedges can join a subset of vertices rather than only two vertices. The vertex coloring of a hypergraph is said to be ”proper” if any hyperedge containing two or more vertices contains at least two distinct colors[12].

References

1. Ahmed, S. (2012). Applications of graph coloring in modern computer science. International Journal of Computer and Information Technology, 3(2), 1-7.
2. Malaguti, E., Monaci, M., & Toth, P. (2011). An exact approach for the vertex coloring problem. Discrete Optimization, 8(2), 174-190.
3. San Segundo, P. (2012). A new DSATUR-based algorithm for exact vertex coloring. Computers & Operations Research, 39(7), 1724-1733.
4. Al-Omari, H., & Sabri, K. E. (2006). New graph coloring algorithms. American Journal of Mathematics and Statistics, 2(4), 739-741.
5. Malaguti, E., & Toth, P. (2010). A survey on vertex coloring problems. International transactions in operational research, 17(1), 1-34.

6. Barenboim, L., & Elkin, M. (2011). Deterministic distributed vertex coloring in polylogarithmic time. Journal of the ACM (JACM), 58(5), 1-25.
7. Malaguti, E., Monaci, M., & Toth, P. (2008). A metaheuristic approach for the vertex coloring problem. INFORMS Journal on Computing, 20(2), 302-316.
8. Finocchi, I., Panconesi, A., & Silvestri, R. (2005). An experimental analysis of simple, distributed vertex coloring algorithms. Algorithmica, 41(1), 1-23.
9. Sudholt, D., & Zarges, C. (2010, December). Analysis of an iterated local search algorithm for vertex coloring. In International Symposium on Algorithms and Computation (pp. 340-352). Springer, Berlin, Heidelberg.
10. Nogueira, B., Tavares, E., & Maciel, P. (2021). Iterated local search with tabu search for the weighted vertex coloring problem. Computers & Operations Research, 125, 105087.
11. Weiher, M., Herzig, M., Tetzlaff, R., Ascoli, A., Mikolajick, T., & Slesazeck, S. (2021). Improved vertex coloring with NbOx memristor-based oscillatory networks. IEEE Transactions on Circuits and Systems I: Regular Papers, 68(5), 2082-2095.
12. Samanta, S., Lee, J. G., Naseem, U., Khan, S. K., & Das, K. (2020). Concepts on Coloring of Cluster Hypergraphs with Application. Mathematical Problems in Engineering, 2020.